**BÁO CÁO SƠ BỘ BÀI TẬP LỚN SỐ 2 MÔN HỆ CƠ SỞ DỮ LIỆU**

**HK231 - Lớp: L02**

**Nhóm**: DBS\_232\_ABC

**Danh sách thành viên**:

|  |  |  |
| --- | --- | --- |
| Stt | Họ tên | MSSV |
| 1 | Trần Nguyễn Thanh Lâm | 2211822 |
| 2 | Nguyễn Gia Thịnh | 2213286 |
| 3 | Nguyễn Đăng Khoa | 2211621 |
| 4 | Nguyễn Hữu Đăng Khoa | 2211625 |
| 5 | Lê Thành Đạt | 2210683 |

**Chủ đề**: E-learning: hiện thực cơ sở dữ liệu phục vụ cho trang E-learning dành cho sinh viên và giáo viên.

**1. Thiết kế ERD cập nhật mới nhất**

<Hình ERD> (*Có thể in A3 đính kèm theo giống báo cáo BTL1*)

**2. Trigger – Thủ tục - Hàm**

**2.1 Thủ tục INSERT / UPDATE / DELETE dữ liệu vào 1 bảng dữ liệu**

Bảng dữ liệu: Professor

1. Thủ tục INSERT:

CREATE OR ALTER PROCEDURE Insert\_professor(

  @ProfID VARCHAR(9),

  @mail VARCHAR(255),

  @name VARCHAR(255),

  @DoB DATE,

  @sex VARCHAR(10),

  @password VARCHAR(10),

  @Degree VARCHAR(255)

)

AS

BEGIN

  IF NOT EXISTS (SELECT 1 FROM UserTable WHERE userID = @ProfID)

  BEGIN

  PRINT 'Chua co giang vien nay, tao giang vien moi';

  INSERT INTO UserTable (userID, mail, name, DoB, sex, password)

  VALUES (@ProfID, @mail, @name, @DoB, @sex, @password);

  INSERT INTO Professor (ProfID, Degree)

  VALUES (@ProfID, @Degree);

  END

  ELSE

  BEGIN

  INSERT INTO Professor (ProfID, Degree)

  VALUES (@ProfID, @Degree);

  END

END;

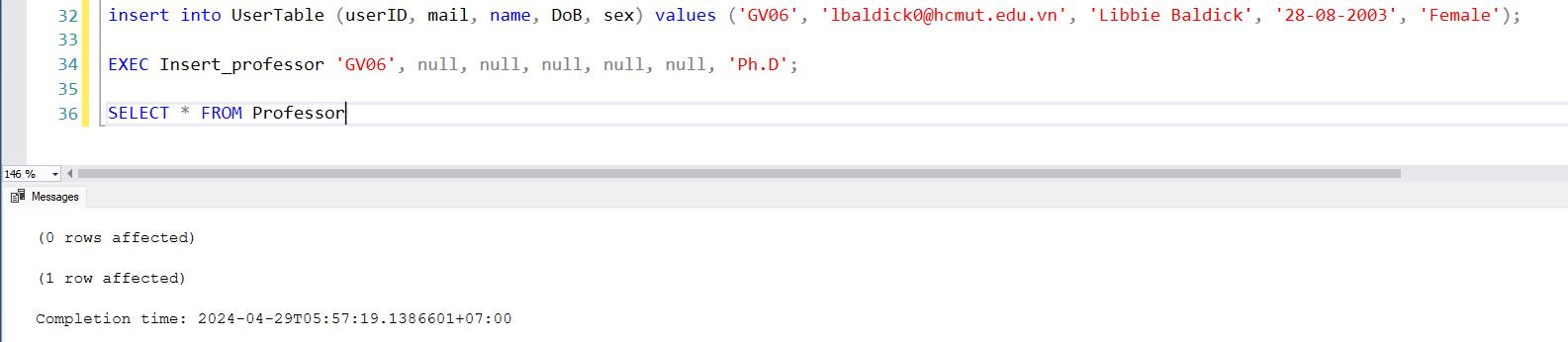
Mô tả thủ tục: Thủ tục INSERT trên được sử dụng để thêm một giảng viên với vào bảng.

Use case: Khi có giảng viên mới cần thêm.

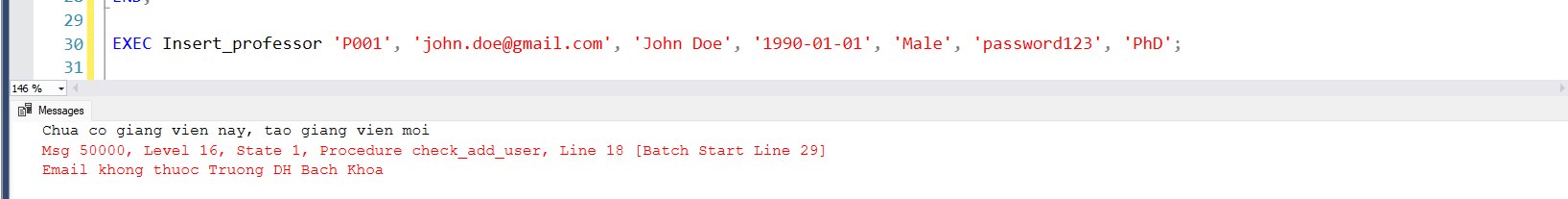
Thao tác chính:

- Kiểm tra xem giảng viên mới đó đã có trong bảng người dùng UserTable hay chưa, nếu chưa thêm vào bảng Usertable. (kiểm tra thông tin người dùng có hợp lệ không).

- Thêm vào bảng Professor.

Input - Output 1: 

Input - Output 2:



1. Thủ tục UPDATE:

CREATE PROCEDURE UpdateProfessorDegree(

  @ProfID VARCHAR(9),

  @NewDegree VARCHAR(255)

)

AS

BEGIN

  UPDATE Professor

  SET Degree = @NewDegree

  WHERE ProfID = @ProfID;

END;

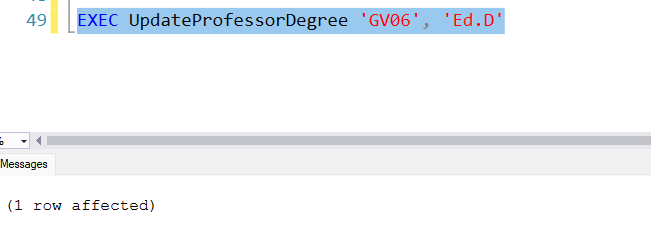
Mô tả thủ tục: Thủ tục UPDATE trên được sử dụng để cập nhật bằng cấp cao hơn cho giảng viên.

Use case: Khi giảng viên đạt bằng cấp cao hơn.

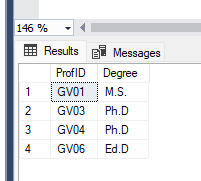
Thao tác chính:

- Cập nhật bằng cấp với mã giảng viên được nhập.

Input:



Output:



1. Thủ tục DELETE:

CREATE OR ALTER PROCEDURE DeleteUnusedProfessor

AS

BEGIN

  DELETE FROM Professor

  WHERE ProfID NOT IN (SELECT ProfID FROM Class)

  DELETE FROM UserTable

  WHERE userID NOT IN (SELECT ProfID FROM Class)

END;

Mô tả thủ tục: Thủ tục DELETE trên được sử dụng để xóa các Giảng viên không tham gia dạy bất kì lớp nào.

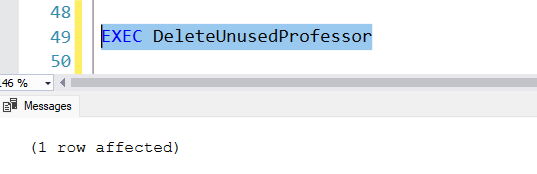
Use case: Sử dụng khi xóa các giảng viên thừa, không tham gia vào giảng dạy.

Thao tác chính:

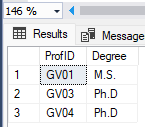
- Xác định điều kiện xóa là ProfID không giảng dạy bất kì lớp nào trong bảng Class.

- Xóa giảng viên lần luôt ở bảng Professor và bảng UserTable.

Input:



Output:



**2.2 TRIGGER**

1. Trigger 1 (hoặc Nhóm các Trigger 1)

CREATE TRIGGER UpdateClassSize

ON Study

AFTER INSERT, DELETE, UPDATE

AS

BEGIN

    IF EXISTS (SELECT 1 FROM inserted) OR EXISTS (SELECT 1 FROM deleted)

    BEGIN

        UPDATE c

        SET Class\_size = s.StudentsCount

        FROM Class c

        INNER JOIN (

            SELECT ClassID, COUNT(StuID) AS StudentsCount

            FROM Study

            GROUP BY ClassID

        ) s ON c.ClassID = s.ClassID;

    END

END;

Mô tả trigger: Trigger dùng để cập nhật sĩ số của lớp.

Usecase: Khi một bảng lớp được thay đổi (thêm, xóa, cập nhật) trigger được kích hoạt để cập nhật sĩ số của lớp.

Thao tác chính: Khi tiến hành thêm, xóa, cập nhật giá trị cho bảng “Study”, tiến hành đếm số lượng sinh viên có cùng “ClassID” sau đó trả về giá trị “Class\_size” cho bảng “Class”

1. Trigger 2

CREATE OR ALTER TRIGGER trg\_UpdateFinalScore

ON StuWork

AFTER INSERT

AS

BEGIN

    DELETE FROM FinalScore WHERE StuID = (SELECT StuID FROM inserted) AND TestID = (SELECT TestID FROM inserted)

    INSERT INTO FinalScore (StuID, TestID, TimesID, Score)

    SELECT

        s.StuID,

        s.TestID,

        s.TimesID,

        s.Score

    FROM StuWork s

    INNER JOIN (

        SELECT StuID, TestID, MAX(Score) AS MaxScore

        FROM StuWork

        WHERE StuID = (SELECT StuID FROM inserted) AND TestID = (SELECT TestID FROM inserted)

        GROUP BY StuID, TestID

    ) max\_scores

    ON s.StuID = max\_scores.StuID

    AND s.TestID = max\_scores.TestID

    AND s.Score = max\_scores.MaxScore;

    WITH WeightedScores AS (

        SELECT

            Fs.StuID,

            c.ClassID,

            COALESCE(SUM(Fs.Score \* Mc.Percentage), 0) / 100 AS Avg\_Score

        FROM FinalScore AS Fs

        JOIN Test AS t ON Fs.TestID = t.TestID

        JOIN Class AS c ON t.ClassID = c.ClassID

        JOIN MarkColumns AS mc ON c.CourseID = mc.CourseID AND t.MarkName = mc.MarkName

        GROUP BY Fs.StuID, c.ClassID

    )

    UPDATE s

    SET Avg\_Score = WeightedScores.Avg\_Score

    FROM Study AS s

  JOIN WeightedScores ON s.StuID = WeightedScores.StuID AND s.ClassID = WeightedScores.ClassID;

END;

Mô tả trigger: Trigger dùng để cập nhật điểm trung bình môn học của mỗi học sinh.

Usecase: Sau khi sinh viên hoàn thành đủ các bài kiểm tra sẽ trả về điểm trung bình của môn học, nếu không trả về giá trị “NULL”

Thao tác chính:  
 - Tạo bảng “FinalScore” là bảng lưu trữ lần làm bài cao điểm nhất của sinh viên.  
 - Tạo bảng “WeightedScores” là bảng dùng để tính điểm trung bình của môn học dựa trên phần trăm điểm của từng bài kiểm tra mà môn học quy định (Cột “Percentage” của “MarkColumns”)  
 - Trả về giá trị “Avg\_Score” của bảng “WeightedScores” cho bảng “Study”

1. Trigger 3

CREATE OR ALTER TRIGGER check\_add\_user

ON UserTable

INSTEAD OF INSERT

AS

BEGIN

    DECLARE @ErrorMessage NVARCHAR(100)

    IF NOT EXISTS (SELECT 1 FROM INSERTED WHERE INSERTED.sex LIKE 'Male' OR INSERTED.sex LIKE 'Female')

    BEGIN

        SET @ErrorMessage = 'Gioi tinh khong hop le'

        RAISERROR (@ErrorMessage, 16, 1)

        RETURN

    END

    IF NOT EXISTS (SELECT 1 FROM INSERTED WHERE INSERTED.mail LIKE '%@hcmut.edu.vn')

    BEGIN

        SET @ErrorMessage = 'Email khong thuoc Truong DH Bach Khoa'

        RAISERROR (@ErrorMessage, 16, 1)

        RETURN

    END

    INSERT INTO UserTable (userID, mail, name, DoB, sex, password)

    SELECT INSERTED.userID, INSERTED.mail, INSERTED.name, INSERTED.DoB, INSERTED.sex, INSERTED.password FROM INSERTED

END

insert into UserTable (userID, mail, name, DoB, sex) values ('GV01', 'lbaldick0@hcmut.edu.vn', 'Libbie Baldick', '28-08-2003', 'Female');

Mô tả Trigger: khi thêm người dùng mới, Trigger kiểm tra giới tính, email của người dùng, nếu hợp lệ sẽ thêm vào bảng UserTable.

Usecase: Trigger được kích hoạt bất kì khi nào có dữ liệu mới được INSERT vào bảng UserTable nhằm kiểm tra tính hợp lệ của dữ liệu.

Thao tác chính:

- Xác định bảng mục tiêu và các giá trị thêm vào: insert into UserTable (userID, mail, name, DoB, sex) values

- Lần lượt kiểm tra giới tính chỉ nhận 2 giá trị là “Male” hoặc “Female” và email phải có cụm [“@hcmut.edu.vn”](mailto:\“@hcmut.edu.vn\”) thuộc trường ĐH Bách Khoa.

- Nếu vi phạm, trả ra lỗi vào biến @ErrorMessage, ngược lại nhập giá trị vào bảng UserTable.

**2.3 THỦ TỤC**

1. Thủ tục 1

CREATE PROCEDURE arange\_student\_on\_score(@ClassID INT)

AS

BEGIN

    SELECT

        s.StuID,

        s.Avg\_Score,

        u.name

    FROM

        Study s

  JOIN

        UserTable u ON u.userID = s.StuID

  JOIN

        Class c ON c.ClassID = @ClassID

    WHERE s.ClassID = c.ClassID

    ORDER BY

        s.Avg\_Score DESC;

END;

Mô tả thủ tục: Thủ tục dùng để truy xuất 1 bảng các học sinh của một lớp theo thứ tự điểm trung bình từ cao đến thấp

Usecase: Sắp xếp học sinh theo thứ tự điểm trung mình của môn học đó.

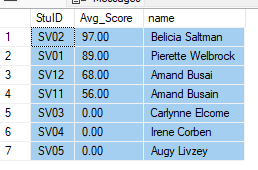
Thao tác chính:  
 - Chọn các sinh viên học lớp @ClassID, sau đó in ra bảng được sắp xếp theo thứ tự điểm trung bình.

Input:

EXECUTE [dbo].[arange\_student\_on\_score]

  @ClassID = 1;

Output:



1. Thủ tục 2

CREATE OR ALTER PROCEDURE GetScholarshipStudents

    @SemesterID INT

AS

BEGIN

    SET NOCOUNT ON;

    DECLARE @TotalStudents INT;

    SELECT @TotalStudents = COUNT(DISTINCT s.StuID)

    FROM Study s

    JOIN Class c ON s.ClassID = c.ClassID

    WHERE c.SemesterID = @SemesterID;

    WITH StudentStats AS (

      SELECT

        s.StuID,

        [dbo].[GetStudentSemesterAverageGrade](s.StuID, @SemesterID) AS AverageGrade,

        COUNT(s.StuID) AS CountClass

      FROM Study s

      JOIN Class c ON s.ClassID = c.ClassID

      WHERE c.SemesterID = @SemesterID

      GROUP BY s.StuID

    ),

    QualifiedStudents AS (

      SELECT

        StuID,

        AverageGrade,

        CountClass

      FROM StudentStats

      GROUP BY StuID, AverageGrade, CountClass

      HAVING AverageGrade > 70

        AND CountClass >= 3

    ),

    TopStudents AS (

      SELECT \*,

        ROW\_NUMBER() OVER (ORDER BY AverageGrade DESC) AS RowNum

      FROM QualifiedStudents

    )

    SELECT StuID, AverageGrade, CountClass, RowNum as Rank

    FROM TopStudents

    WHERE RowNum <= CEILING(0.1 \* @TotalStudents);

END;

Mô tả thủ tục: Thủ tục dùng để lấy danh sách những học sinh được nhận học bổng trong kì này.

Usecase: Khi muốn chọn 10% số học sinh đạt các chỉ tiêu như điểm trung bình > 70 và học nhiều hơn 3 môn trong kì này.

Thao tác chính:

- Chọn các sinh viên theo học trong học kì này, sắp xếp vào bảng QualifiedStudents nếu như sinh viên đó đạt các yêu cầu.

- Chọn 10% số sinh viên trong bảng QualifiedStudents

Input:

EXEC GetScholarshipStudents 223

Output:

1. Thủ tục 3

CREATE OR ALTER PROCEDURE check\_student\_submission\_count (@student\_id VARCHAR(9), @classID INT)

AS

BEGIN

  DECLARE @submission\_count INT

  DECLARE submission\_cursor CURSOR FOR

    SELECT COUNT(\*) AS submission\_count

    FROM StuWork AS sw

  JOIN Class c ON c.ClassID = @classID

    WHERE sw.StuID = @student\_id;

  OPEN submission\_cursor;

  FETCH NEXT FROM submission\_cursor INTO @submission\_count;

  CLOSE submission\_cursor;

  DEALLOCATE submission\_cursor;

  IF @submission\_count IS NULL

  BEGIN

    SELECT 'Sinh vien chua nop bai nao'

  END

  ELSE

  BEGIN

    SELECT CONCAT('Sinh vien da nop ', @submission\_count, ' bai')

  END;

END;

Mô tả thủ tục: Thủ tục dùng để kiểm tra sinh viên đã nộp bao nhiêu bài thi trong lớp.

Usecase: Khi kiểm tra sinh viên đã nộp bao nhiêu lần làm bài tổng cộng trong tất cả bài kiểm tra

Thao tác chính:

- Tạo vòng lặp, sau đó đếm số bài mà sinh viên đã nộp

- Xuất ra số bài mà sinh viên đã nộp

Input:

EXECUTE [dbo].[check\_student\_submission\_count]

  @student\_id = 'SV11',

  @classID = 1;

Output:



1. Thủ tục 4

CREATE OR ALTER PROCEDURE UpdateMissingStuWorkScores (@CurDATE DATETIME)

AS

BEGIN

    WITH TestThatStudentDontDo AS (

        SELECT tn.StuID, tn.TestID, Fs.Score

        FROM (

            SELECT StuID, t.TestID, t.Deadline

            FROM Study AS st

            JOIN Test AS t ON st.ClassID = t.ClassID

        ) AS tn

        LEFT JOIN (

            SELECT StuID, TestID, MAX(Score) AS Score

            FROM StuWork

            GROUP BY StuID, TestID

        ) AS Fs ON Fs.StuID = tn.StuID AND Fs.TestID = tn.TestID

        WHERE tn.Deadline < @CurDATE

    )

    INSERT INTO StuWork (StuID, TestID, TimesID, Score)

    SELECT tnd.StuID, tnd.TestID, 1, 0

    FROM TestThatStudentDontDo AS tnd

    WHERE tnd.Score IS NULL;

END;

Mô tả thủ tục: Thủ tục dùng để tạo ra các bài làm 0 điểm cho các sinh viên không hoàn thành bài thi khi đã hết Deadline.

Usecase: Để tạo các bảng ghi 0 điểm cho sinh viên để tính điểm trung bình.

Thao tác chính:

- Kiểm tra các bài làm cũ của sinh viên, nếu sinh viên chưa làm, kiểm tra nếu Deadline đã quá @CurDATE.

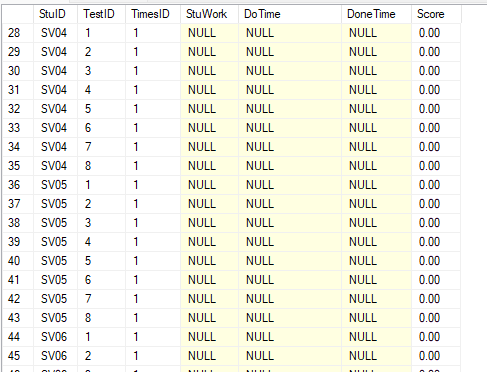
- Thêm các giá trị mới vào bảng StuWork nếu sinh viên chưa thực hiện bài thi.

Input:

EXECUTE [dbo].[UpdateMissingStuWorkScores]

  @CurDATE = '2024-27-04'

Output:



1. Thủ tục 5

REATE PROCEDURE UpdateClassStatus(@CurDATE DATETIME, @SemesterID INT)

AS

BEGIN

  IF @CurDATE BETWEEN (SELECT StartDate FROM Semester WHERE SemesterID = @SemesterID)

                     AND (SELECT EndDate FROM Semester WHERE SemesterID = @SemesterID)

    BEGIN

         SELECT 'Ongoing'

    END

  ELSE

  BEGIN

    UPDATE Class

    SET Status = 'closed'

    WHERE SemesterID = @SemesterID;

  END

END;

Mô tả thủ tục: Thủ tục UPDATE trên được sử dụng để cập nhật trạng thái của tất cả các lớp trong học kì.

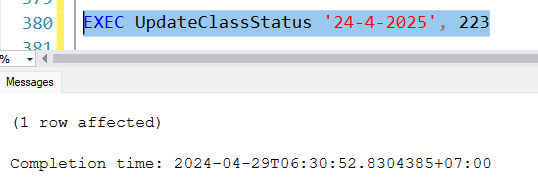
Use case: Khi học kì đã kết thúc, cần cập nhật trạng thái của lớp học đã đóng.

Thao tác chính:

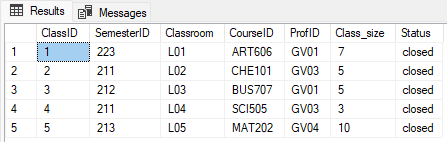
- Kiểm tra biến @CurDate được nhập vào, nếu nằm giữa 2 giá trị StartDate và EndDate của học kì trùng với biến @SemesterID thì trả về dòng chữ “Ongoing”.

- Ngược lại, tiến hành cập nhật tất cả các lớp thuộc học kì đó có Status là “closed”.

Input:



Output:



**2.4 HÀM**

1. Hàm 1

CREATE OR ALTER FUNCTION CountStudentsAboveScore(@ScoreIN DECIMAL(10,2), @ClassID INT)

RETURNS INT

AS

BEGIN

IF (@ScoreIN > 100)

    BEGIN

    RETURN NULL;

    END;

  DECLARE @Count INT = 0;

  DECLARE @Score DECIMAL(10,2);

    DECLARE myCursor CURSOR FOR

        SELECT Avg\_Score

        FROM Study

        WHERE ClassID = @ClassID;

    OPEN myCursor;

    FETCH NEXT FROM myCursor INTO @Score;

    WHILE @@FETCH\_STATUS = 0

    BEGIN

        IF @Score > @ScoreIN

        BEGIN

            SET @Count = @Count + 1;

        END

        FETCH NEXT FROM myCursor INTO @Score;

    END;

    CLOSE myCursor;

    DEALLOCATE myCursor;

    RETURN @Count;

END;

Mô tả hàm: Hàm đếm số sinh viên nhiều hơn số điểm đã nhập trong 1 lớp

Usecase: Khi kiểm tra số lượng sinh viên đạt yêu cầu trên số điểm nhất định

Thao tác chính:

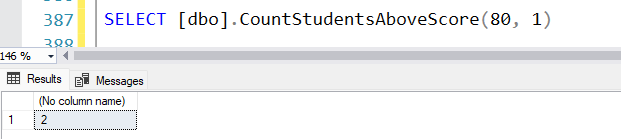
- Kiếm tra nếu điểm nhập vào lớn hơn 100 sẽ trả về NULL.

- Tạo con trỏ mang giá trị điểm trung bình myCursor và biến @Count để đếm.

- Với mỗi sinh viên có điểm trung bình cao hơn yêu cầu, tăng biến @Count lên 1.

- Trả về giá trị @Count.

Input - Output:



1. Hàm 2

CREATE OR ALTER FUNCTION GetStudentSemesterAverageGrade(@StuID VARCHAR(9), @SemesterID INT)

RETURNS DECIMAL(10,2)

AS

BEGIN

    DECLARE @StudentGrades TABLE(

        CourseID CHAR(6),

        Avg\_Score DECIMAL(10,2)

    );

    INSERT INTO @StudentGrades

    SELECT CourseID, Avg\_Score

    FROM Study s JOIN Class c ON s.ClassID = c.ClassID

    WHERE StuID = @StuID AND SemesterID = @SemesterID;

    DECLARE @AverageGrade DECIMAL(10,2);

    SELECT @AverageGrade = AVG(Avg\_Score)

    FROM @StudentGrades;

    RETURN @AverageGrade;

END;

Mô tả hàm: Hàm trả về điểm trung bình học kì của sinh viên

Usecase: Khi tổng kết cuối kì

Thao tác chính:

- Tạo bảng phụ @StudentGrades mang giá trị Avg\_Score ứng với từng CourseID

- Khai báo biến @AverageGrade và nhập điểm trung bình vào.

Input - Output:

